Generation and analysis of networks with a prescribed degree sequence and subgraph family: higher-order structure matters

Article (Published Version)

Journal of Complex Networks (2017) **5**, 1–31
doi:10.1093/comnet/cnw011
Advance Access Publication on 16 May 2016

Generation and analysis of networks with a prescribed degree sequence and subgraph family: higher-order structure matters

MARTIN RITCHIE
School of Mathematical and Physical Sciences, Department of Mathematics, University of Sussex, Falmer, Brighton BN1 9QH, UK

LUC BERTHOUZE
Centre for Computational Neuroscience and Robotics, University of Sussex, Falmer, Brighton BN1 9QH, UK

AND

ISTVAN Z. KISS[†]
School of Mathematical and Physical Sciences, Department of Mathematics, University of Sussex, Falmer, Brighton BN1 9QH, UK
[†]Corresponding author: Email: I.Z.Kiss@sussex.ac.uk

Edited by: Ernesto Estrada

[Received on 26 November 2015; accepted on 9 March 2016]

Designing algorithms that generate networks with a given degree sequence while varying both subgraph composition and distribution of subgraphs around nodes is an important but challenging research problem. Current algorithms lack control of key network parameters, the ability to specify to what subgraphs a node belongs to, come at a considerable complexity cost or, critically and sample from a limited ensemble of networks. To enable controlled investigations of the impact and role of subgraphs, especially for epidemics, neuronal activity or complex contagion, it is essential that the generation process be versatile and the generated networks as diverse as possible. In this article, we present two new network generation algorithms that use subgraphs as building blocks to construct networks preserving a given degree sequence. Additionally, these algorithms provide control over clustering both at node and global level. In both cases, we show that, despite being constrained by a degree sequence and global clustering, generated networks have markedly different topologies as evidenced by both subgraph prevalence and distribution around nodes, and large-scale network structure metrics such as path length and betweenness measures. Simulations of standard epidemic and complex contagion models on those networks reveal that degree distribution and global clustering do not always accurately predict the outcome of dynamical processes taking place on them. We conclude by discussing the benefits and limitations of both methods.

Keywords: networks; clustering; subgraphs; epidemics; complex contagion.

1. Introduction

Being able to replicate, and therefore investigate, the structure and function of real-world complex networks is a profoundly difficult problem. However, the pervasiveness of systems that could be more

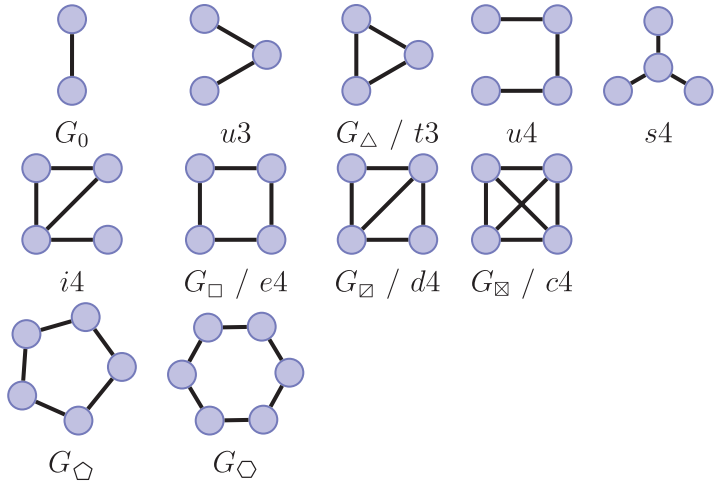

FIG. 1. The set of subgraphs that have been used in this article. The subgraphs denoted by: $\{G_0, G_\triangle, G_\square, G_\boxslash, G_\boxtimes, G_\pentagon, G_\hexagon\}$, are those that have been used as input for the proposed network construction algorithms. We use: $\{u3, t3, u4, s4, i4, e4, d4, c4\}$, to denote the total number of uniquely counted subgraphs given by the subgraph counting algorithm [12].

accurately interpreted as a result cannot be overstated: social networks [1], the spread of disease [2], artificial intelligence [3], language structure [4] and transportation networks [5]. Accordingly, a number of network models and network generating algorithms have been proposed [6–14]. Many of these network models seek to reproduce a specific network property or characteristic: the degree distribution [7, 15, 16], the small worldness [6], degree–degree correlations [2, 17, 18] or clustering, the propensity of three-cycles in a network [19, 20]. However, investigations of *higher-order* structure, subgraphs and arrangements of subgraphs not specified by standard network metrics, have been limited by a lack of accurate and versatile network models. Some progress has been made using the *configuration model* [12, 13, 21], and it is this work we seek to build upon.

In the standard configuration model, triangle subgraphs appear infrequently as a by-product of working with finite size networks [22]. But what if one *wants* triangle subgraphs to appear in a network, in particular, if one wants to model a complex network with clustering? An extension of the configuration model to this case exists [23, 24]. In this extension a node is allocated a number of stubs, that may go on to form standard edges, as well as a number of triangle 'corners' or *hyperstubs*, pairs of stubs that will form triangles. While edges are formed in the usual way, triangles are formed by selecting three triangle hyperstubs at random and connecting their pairs of constituent stubs.

As for edges, the number of all stubs must be divisible by two, the total number of triangle hyperstubs must be divisible by three is a necessary condition for the triangle hyperstub sequence to be graphical. Another similarity this model shares with the standard configuration model is that the probability that any two triangles will share an edge, thus forming a $G_\boxslash$ subgraph (see Fig. 1), vanishes in the limit of large network size [21]. Just as a network composed of lines only is limited in recreating real-world networks, so is a model that can only include edges and triangles. Obviously, this may depend on properties and structure of the real networks, but in many cases, edges and triangles are not enough to produce an accurate enough artificial replica of the real network.

The configuration model has since received further attention to address this [21]. Building on the edge-triangle model, a more general subgraph-based approach is taken where one may specify distributions of edges alongside distributions of arbitrary subgraphs. In the case of complete subgraphs it is obvious how to do this. For example, $G_{\boxtimes}$ subgraphs can be formed by allocating to nodes hyperstubs composed of three stubs. Then, four of these hyperstubs can be selected at random to form a $G_{\boxtimes}$ subgraph. However, it is not clear how this may work for subgraphs that are composed of more than one type of hyperstubs. For example, in a $G_{\boxslash}$, there are two different types of hyperstubs, and it is necessary for any network model or construction algorithm to be able to make this distinction. Karrer and Newman proposed that it is possible to identify a node's *role* within a subgraph using *orbits*. To find the orbits of a subgraph one must first list all possible automorphisms of the subgraph, that is, permutations of nodes that do not create or destroy edges. The orbit of a node is a set of other nodes with which it may be permuted so that no edges are created or destroyed. Of course, computing the automorphism group of subgraphs is computationally challenging but so long as subgraphs with few nodes are used, this is not a problem [21].

Network models are rarely used independently of other processes. Instead, they typically provide the substrate for dynamical processes to operate upon. For example, the compartmental susceptible-infected-recovered (*SIR*) model of contagion is often embedded into a network to help better understand how the network and its properties affect the epidemic. Previous work [13] successfully incorporated the Karrer and Newman approach into an approximate ordinary differential equations (ODE) or mean-field model for *SIR* epidemics on networks displaying higher-order structure, and this mean-field model showed excellent agreement with simulation results. In order to achieve this, Ritchie et al. bypassed the need to classify a node's role in a subgraph via the automorphism group. Instead, nodes within arbitrary subgraphs were uniquely enumerated, even if they were topologically equivalent to one another, and this enumeration defined their role. The motivation for this adaptation was to simplify the derivation of the ODE model. Using the orbit approach or the full enumeration are different ways of satisfying different modelling needs, and these are not the only possible approaches. In fact, when modelling networks and nodes within subgraphs, one can instead classify nodes by the stub cardinality of their hyperstubs.

A common method across all of the above models, i.e., edge-triangle, the more general Karrer–Newman model, and that proposed by Ritchie et al., is that sequences of hyperstubs must be specified for each and every subgraph that is to be included. From these sequences it is possible to recover the network's degree sequence by multiplying them by the stub cardinality of the hyperstub which they represent and then summing the resulting sequences. Therefore the degree sequence of the network is a result of the construction of the network rather than a quantity that is controlled for. However, given that the degree sequence of the network is probably the single most important characteristic of a network, there is a need for methods that can generate networks with a particular subgraph family and distribution yet preserve a given degree sequence. In [13], we recently showed that it is possible to constrain the hyperstub sequences so that the 1st and 2nd moments of the resulting degree sequence are controlled. In this article, we go beyond this work and propose two generation algorithms that provide full control over the degree sequence and clustering.

The article is organized as follows. In Section 2, we describe in detail the two generation algorithms, including tuning of clustering. In Section 3, we validate our algorithms and we explore the diversity of the generated networks by comparing them to the widely used Big-V rewiring scheme. We further analyse networks generated by using different subgraph families or distributions. Epidemic and complex contagion models are simulated on these networks, and we show that degree distribution and global clustering alone are not sufficient to predict the outcome of these processes. Finally, we discuss extensions and further research questions relating to our work.

2. Materials and methods

In this section we propose two new algorithms, both of which are parametrized by a degree sequence and a set of subgraphs. The algorithms construct hyperstub degree sequences (from which the input degree sequence may be recovered exactly) that can be used in a modified configuration model style connection procedure to realize a network.

There are some caveats regarding the preservation of the input degree sequence that are common to all configuration-like models. First it is necessary for a degree sequence to sum to an even number to be graphical. If it does not, a stub must be created or destroyed to satisfy this constraint. In general, hyperstub degree sequences must sum with multiplicity equal to the number of times they appear in their parent subgraphs, i.e., a triangle hyperstub sequence must be divisible by 3. When selecting stubs or hyperstubs at random to form subgraphs, it is possible that self or multi-edges may form. The number of these events happening depends only on the average degree $\langle k \rangle$ and thus remains constant with network size. It is possible to simply delete self-edges or collapse multi-edges down to a single edge. If this approach is taken then the guiding degree sequence will be violated. Instead we disallow such connections by reselecting nodes in the connection procedure until no self or multi-edges will be created by forming the subgraph. This is known as the *matching algorithm* [25]. Finally, it is possible for the process to be left with no option other than to add subgraphs over existing links or selecting multiple instances of the same node. In this case we completely reset the algorithm, regenerating hyperstub sequences and forming subsequent connections until a network is formed.

2.1 *The underdetermined sampling algorithm*

The concept underpinning this algorithm is that for each node there are combinations of hyperstubs that will satisfy its degree. For example, a node with $k = 3$ classical edges could form 3 single G_0 edges or 1 G_0 edge and 1 $G_\triangle$ hyperstub. The number of possible arrangements will depend on the degree of the node and number of input subgraphs. From these arrangements a single one is selected at random. For a given degree k, this problem is equivalent to solving an underdetermined linear Diophantine equation equal to k subject to positivity constraints. The coefficients are given by the edge counts of the hyperstubs, that are induced by the input subgraphs, and the solution will give the number of each hyperstub so that the degree of the node is matched exactly.

To generate a network using this algorithm, let us assume that a degree sequence, $D = \{d_1, d_2, \ldots, d_N\} \in \mathbb{N}_0^{1\times N}$, and the set of subgraphs to be included in the network's construction, $G = \{G_1, G_2, \ldots, G_l\}$, is given. Then, for each subgraph we classify its hyperstubs by their edge cardinality. It is now possible to form a vector that has elements specifying the number of edges in each hyperstub. From this vector we take the unique elements. For example, the $G_{\boxslash}$ subgraph will have a corresponding hyperstub vector of $\alpha = (2, 3)$. For a given degree k we must consider all possible hyperstubs and hyperstub combinations that yield a classical degree equal to k. To systematically list all such combinations, we first concatenate all the hyperstub vectors into a single vector, $\boldsymbol{\alpha}$, to be used as coefficients for the following linear underdetermined Diophantine equation

$$k = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_r x_r, \tag{1}$$

where $k = k_{\min}, k_{\min} + 1, \ldots, k_{\max}$ and r denotes the number of eligible hyperstubs—a node with degree $k = 3$ can only go on to form subgraphs where the hyperstubs contain no more than three edges—for the given degree k and which is solved subject to the constraint $\boldsymbol{x} \in \mathbb{N}_0^r$. A solution $\boldsymbol{x}$ of this equation corresponds to the number of each type of hyperstubs required to result in a node of degree k. For example,

if α_1 and α_2 take values 1 and 2 corresponding to hyperstubs of G_0 and $G_\triangle$, respectively, and the degree of the node is $k = 5$, the Diophantine equation would take the form $5 = x_1 + 2x_2$ and the solution space of this equation is given by the pairs $(x_1, x_2) = \{(5, 0),\ (3, 1),\ (1, 2)\}$. In general these equations may be solved recursively by fixing a trial value $x_i = j$ and reducing the dimensionality of the equation by absorbing this term. This is repeated until the equation becomes of the standard form: $k' = \alpha_1 x_1 + \alpha_2 x_2$, which can be solved explicitly. A solution obtained this way will form a single solution of the original equation. This process is then repeated for a different starting trial solution, and since we seek only positive solutions and k is finite, the corresponding solution space has a finite number of elements. Matlab code for this process is available at https://github.com/martinritchie/Network-generation-algorithms. Accessed on 23 April 2016.

Once the entire solution space for each degree has been found it is possible to start forming the hyperstub degree sequences. To proceed, the algorithm works sequentially through the degree sequence $D = \{d_1, d_2, \ldots, d_N\}$ of the N nodes, where $d_i \in \{k_{\min}, k_{\min} + 1, \ldots, k_{\max}\}$. By selecting at random a solution from the solution space that corresponds to $k = d_i$, that specifies the hyperstub configuration, and by concatenating all the selected solutions for all the nodes a hyperstub degree sequence of dimension $h \times N$, where h denotes the total number of hyperstubs induced by the input subgraphs, is formed.

For incomplete subgraphs it is not possible to select solutions of the Diophantine equations' solution spaces at random. The reason for this is two-fold: (1) not all incomplete subgraphs are composed of equal quantities of each of their constituent hyperstubs and (2) hyperstubs with lower stub cardinality will appear more frequently than hyperstubs of higher stub cardinality because hyperstubs with fewer edges can be more readily accommodated into the degree of a node. Problem (1) may be addressed by representing every hyperstub induced by a subgraph in the vector of coefficients opposed to grouping hyperstubs by their stub cardinality. Problem (2) may be addressed by decomposing hyperstubs generated in excess into simple/classical edges. It should be noted that both of these methods will bias the resulting sequences but that this bias is only present when incomplete subgraphs are specified as input for the undetermined sampling algorithm (UDA). One advantage of the method we use is that it is possible to calculate the number of hyperstubs that will be decomposed back into stubs using *integer partitions*, and this is detailed in Appendix A.1. In particular the following result holds,

$$p(k, \alpha) = \sum_{m=1}^{\lfloor \frac{k}{\alpha} \rfloor} m\big[p(k - \alpha m)) - p(k - \alpha(m + 1))\big],$$

where $p(k, \alpha)$ is the number of times that α appears in the partitions of k, with $\alpha \leq k$.

This may be used to compute the number of times certain hyperstubs appear. Returning to the example of the homogeneous networks with $k = 5$, generated with G_0 and $G_{\boxslash}$ there will be four counts of the double hyperstub generated for every two counts of the triple corner in the partition space, since 5 can be partitioned as

$$\{\{1, 1, 1, 1, 1\}, \{2, 1, 1, 1\}, \{3, 1, 1\}, \{4, 1\}, \{2, 2, 1\}, \{2, 3\}, \{5\}\},$$

and $p(5, 2) = 4$ and $p(5, 3) = 2$. It should be noted that $p(k, \alpha)$ will count how many times α appears in all possible partitions of k. However, since it is not possible for either a double or triple corner to appear with a degree 4 or 5 hyperstub this will not affect the result. This simple number theoretic consideration shows a viable way in which bias can be quantified or measured.

Pseudocode for the UDA algorithm is given in Appendix A.2, and the Matlab code is available from https://github.com/martinritchie/Network-generation-algorithms. Accessed on 23 April 2016.

2.1.1 *A priori clustering calculation* The global clustering coefficient is defined as the ratio between the total number of triangles and the total number of connected triples of nodes $\triangle + \vee$, since each triangle contains three triples of nodes: $C = \frac{\triangle}{\triangle + \vee}$. It should be noted that each unique triangle is counted six times and each unique triple is counted twice. The number of triples incident to a node of degree k is given by $\triangle + \vee = k(k-1)$ since a node will form a triple with every pair of its neighbours and each triple is counted twice. The expected number of triples for a node of degree k is therefore obtained by summing $P(K = k) \times k(k-1)$ over all degrees, where $P(K = k)$ is the probability of finding a node of degree k. The expected number of triangles incident to a node of degree k, $\langle \triangle_k \rangle$, may be obtained from the Diophantine equations' solution space associated with that degree. To do this, one needs to sum all occurrences of triangle corners, regardless of which subgraph they belong to, from that solution space and divide by the number of solutions in that particular solution space, since solutions are selected uniformly at random. Finally we are in a position to compute the expected global clustering coefficient as

$$C = \sum_{k=2}^{k_{\max}} \frac{\langle \triangle_k \rangle}{P(K = k) \times k(k-1).} \tag{2}$$

For example, let us consider the homogeneous network with $k = 5$ and the input subgraphs G_0 and $G_{\boxslash}$. These subgraphs induce the vector of coefficients $\boldsymbol{\alpha} = (1, 2, 3)$ that, for $k = 5$, has the following solution space

$$\begin{array}{rccccc} G_0: & 5 & 3 & 2 & 1 & 0, \\ g_2: & 0 & 1 & 0 & 2 & 1, \\ g_3: & 0 & 0 & 1 & 0 & 1, \end{array}$$

where the rows give the number of each hyperstub, the columns give an individual solution and g_2 and g_3 denote the double and triple hyperstub of $G_{\boxslash}$, respectively. From this we may calculate the expected number of triangles $\langle \triangle_5 \rangle$. In this example we can see that on average for every g_3 corner the UDA will generate two g_2 corners. Since the excess g_2 corners will be decomposed into edges, one observes that g_2 and g_3 will be generated in equal quantities. So the expected number of g_2 is given by the expected number of g_3, e.g., $2/5$ per node. Since g_2 denotes a triangle corner, the number of g_2 corners also gives the total number of triangles, that is uniquely counted and per node. So the expected number of triangle per node is $12/5$, each triangle being counted six times, and this network will have a theoretical global clustering of $C = 0.12$. Computationally, we verify this by generating such networks with $N = 5000$, and find that the number of open triples and triangles is exactly $|\vee| = 100000$ and $|\triangle| = 12120$, resulting in a global clustering of 0.1212, as expected.

2.2 *Cardinality matching*

The cardinality matching algorithm (CMA) requires as input a degree sequence, a set of subgraphs and corresponding *subgraph sequences*, i.e., multiple sequences specifying to which and how many subgraphs nodes belong to. Note that these sequences are not yet allocated to nodes. The algorithm proceeds to allocate hyperstubs of subgraphs to nodes that have a sufficient number of stubs to accommodate the hyperstub degree. The algorithm outputs hyperstub degree sequences, from which the input degree sequence may be recovered exactly. This then can be used to realize a network based on a modification of the configuration model.

To generate a CMA network, one needs to first decide on a degree sequence D, a subgraph set $G = \{G_1, G_2, \ldots, G_l\}$ and a set of subgraph sequences $S = \{S_1, S_2, \ldots, S_l\}$, where $S_j(k)$, with $j = 1, 2, \ldots, l$ and $k = 1, 2, \ldots, N$, gives the number of times a node will be part of a G_j subgraph without specifying the precise hyperstubs that connect the node to a G_j subgraph. Our goal is to map the subgraph sequences into hyperstub sequences that can then be allocated to nodes that can accommodate them. From the hyperstub sequence, it is possible to work out the lower bound on the degree of nodes that can accommodate a specific hyperstub sequence. To complete this mapping one needs to differentiate between complete and incomplete subgraphs.

For complete subgraphs the subgraph sequence is identical to its hyperstub sequence since there is only one way or hyperstub by which a node can connect to such a subgraph. Thus, multiplying the hyperstub degree by the number of edges in the hyperstub will give us the lower bound on the degree of nodes that can accommodate the hyperstub sequence. For incomplete subgraphs the subgraph sequence does not specify how the node connects to the subgraph. Hence, we need to determine how the various hyperstubs are allocated to nodes. To see how to do this, let us consider an arbitrary subgraph G with subgraph sequence S. Given that the subgraph has m distinct hyperstubs, let $p = (p_1, p_2, \ldots, p_m)$ be the vector of probabilities of picking different hyperstubs. We note that the values of p reflects the proportion of each hyperstub found in the subgraph. For example, $G_{\boxslash}$ has two distinct hyperstubs that both appear with multiplicity two, in this case $p = (1/2, 1/2)$. This will ensure that their numbers are balanced and subgraphs can be formed.

Next, using the multinomial distribution corresponding to subgraph G, $M^G(s_i^G, P)$ where s_i^G denotes the subgraph sequence of index i (this is not yet a node label), we pick hyperstub types to transform the subgraph sequence into hyperstub degree. For each s_i^G this will result in a vector of length m specifying the exact number of each hyperstub. It is possible to concatenate all the resulting choices from all multinomial distributions $M^G(s_i^G, p)$, where $i = 1, 2, \ldots, N$ form the following matrix

$$
\begin{array}{c}
 \\
h_1^G \\
h_2^G \\
\vdots \\
h_m^G
\end{array}
\begin{array}{c}
\begin{array}{cccc} s_1^G & s_2^G & \ldots & s_N^G \end{array} \\
\begin{pmatrix}
h_1^G(1) & h_1^G(2) & \ldots & h_1^G(N) \\
h_2^G(1) & h_2^G(2) & \ldots & h_2^G(N) \\
\vdots & \ddots & \ddots & \vdots \\
h_m^G(1) & h_m^G(2) & \ldots & h_m^G(N)
\end{pmatrix}
\end{array}
= H^G,
$$

where $h_i^G(j)$ denotes the number of h_i hyperstubs contributing to the subgraph degree s_j^G. We now need to compute the total number of edges specified by each column of the above matrix or by the hyperstub degree. This is given by $H^G(i) = \sum_{j=1}^{m} |h_j^G| h_j^G(i)$ that denotes the total number of edges required by the subgraph degree s_i^G, and where $|h_j^G|$ represents the number of edges needed to form hyperstub j in subgraph G and $i = (1, 2, \ldots, N)$. This process needs to be repeated for each subgraph to be included in the networks construction, i.e., for each subgraph G_i with subgraph sequence $S^{G_i} = (s_1^{G_i}, s_2^{G_i}, \ldots, s_N^{G_i})$ there is a corresponding H^{G_i} with elements that the algorithm will use as the lower bound on the degree of the nodes that can accept such a selection of hyperstubs.

The algorithm then proceeds by choosing the largest values, $H_{\max}$, from all H^{G_i} matrices, and this is used as the lower bound on the degree of nodes that can accept the hyperstub configuration associated with $H_{\max}$, i.e., have enough edges of the classical type. From this list of all nodes with degree equal to or larger than $H_{\max}$, a node is selected uniformly at random. The degree of the selected node is reduced

accordingly, and the index of the node is now associated with the hyperstub degree to $H_{\max}$. This node is then removed from the pool of eligible nodes for that particular subgraph, as otherwise it may be selected twice for the same subgraph thus violating the subgraph degree sequence. Similarly, the element $H_{\max}$ is also removed from the pool of subgraph degree sequences that have yet to be allocated to nodes. This needs to be repeated until all elements of each subgraph degree sequence are allocated to nodes. Any edges that are not allocated to a particular hyperstub or subgraph are left to form edges.

In some cases it may be necessary to impose some cardinality constraints on the subgraph sequences. Obviously, if the network is homogeneous with $k = 3$ we cannot include complete pentagon subgraphs or allocate two $G_{\triangle}$ subgraphs to each node. More generally, it may be necessary to constrain the moments of the subgraph sequences. Let $\langle k \rangle$ denote the mean degree of the given degree sequence and let G_i be a subgraph composed of a single hyperstub with cardinality α and having subgraph degree sequence with mean $\langle s \rangle$ then: $\langle \alpha s \rangle = \alpha \langle s \rangle \leq \langle k \rangle$ is a necessary condition for the two sequences to be graphical. In the case of more than one hyperstub, this is extended to $\sum_{i=1}^{m} \alpha_i \langle s_i \rangle \leq \langle k \rangle$, where m, α_i and s_i denote the number of hyperstubs, hyperstub cardinality and associated subgraph sequence, respectively. For the networks generated in this article, the degree sequence and subgraph sequences were measured from networks previously generated by the UDA such that prior knowledge about the sequences being graphical was available without the need to impose any such constraints.

Clustering calculations for this algorithm are trivial since the subgraph degree sequences are known. One simply sums a sequence and then multiplies this figure by the number of triangles induced by that subgraph, being careful not to double count across multiple sequences for the same subgraph. The number of triples of connected nodes can be calculated following the method given for the UDA given in Section 2.1.1. Pseudocode for the CMA is given in Appendix A.3, with the corresponding Matlab code available from https://github.com/martinritchie/Network-generation-algorithms. Accessed on 23 April 2016.

2.3 *Connection process*

We describe this process for a single incomplete subgraph. The case of the complete subgraph is trivial and has already been described (see Section 1). This process was first presented by Karrer and Newman [21]. Consider a subgraph composed of three different hyperstub types, h_1, h_2 and h_3 that occur with a multiplicity of 1, 2 and 3, respectively, i.e., the subgraph is composed of six nodes. We require the following necessary conditions for the hyperstub sequences to be graphical

$$\sum_{i=1}^{N} |h_1|_i = \frac{1}{2} \sum_{i=1}^{N} |h_2|_i = \frac{1}{3} \sum_{i=1}^{N} |h_3|_i, \tag{3}$$

where $|h_i|_j$ specifies the h_i hyperstub degree of node j. If these conditions are not met, one needs to decompose any surplus hyperstubs into stubs that may form classical edges in order to preserve the degree sequence.

Using the hyperstub sequences, one can create three dynamic lists for the three hyperstub types where a node appears with multiplicity equal to its hyperstub degree. Once the dynamic lists are fully populated, the connections process can start. This is done by selecting the following: 1 node from the h_1 bin, 2 from the h_2 bin and 3 from the h_3 bin, and all the selection processes done uniformly at random and without replacement. Before forming the connections between these six nodes, one must ensure that (1) the selection contains no duplicates (that will form self-edges) and (2) that no single pair of nodes are already connected. If a connection already exists, a multi-edge may form and/or

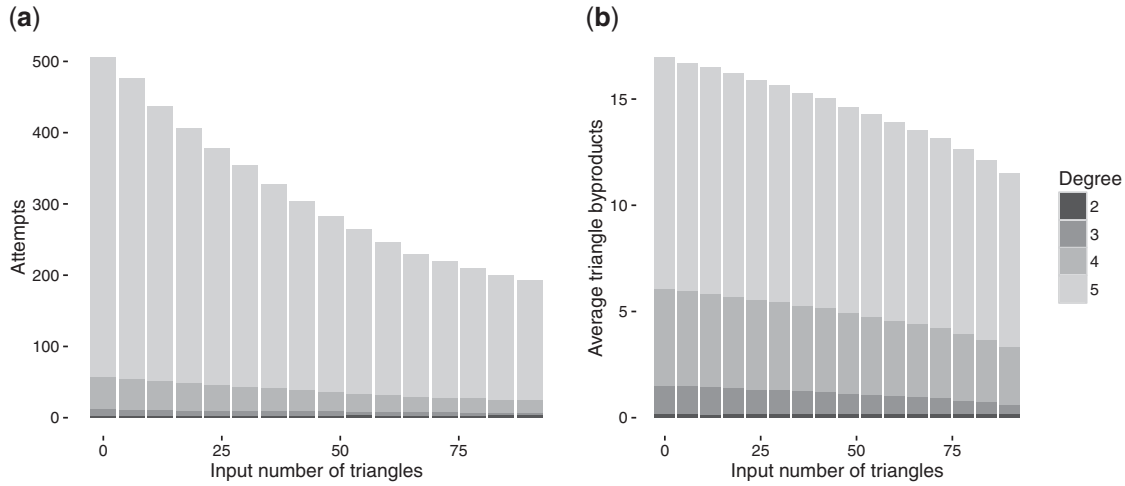

FIG. 2. (a) The average number of *refusal* attempts to realize a network and (b) the average number of triangle by-products found per networks. In both cases the CMA was parametrized with only G_0 and $G_\triangle$ subgraphs. As more $G_\triangle$ are specified as input, both the number of average number of attempts and triangle by-products decreases. Triangle by-products are computed by subtracting the input from the measured number of triangles.

subgraphs will share edges. If neither of these conditions are violated then the connections may be formed. Otherwise, all nodes are returned to their bins and a new selection is made. It is possible that after many selections no valid combinations of nodes remain. For example, all bins may contain the same node. In this and other non-viable cases, all bins are re-populated and the connection process is started anew.

As previously discussed, it is possible to delete self and multi-edges but this will destroy the degree sequence. The method of reselecting nodes has been previously introduced and is known as the *matching algorithm* [25]. However, it has previously been shown that the matching algorithm introduces a bias when constructing networks [26]. Ideally, when a self or multi-edge is formed one would start the whole connection process from scratch, the so-called *refusing algorithm*. This results in an unbiased sampling [26]. For the configuration model the number of such self and multi-edges depends on the first and second moments of the degree distribution [2]. As such, an unbiased configuration model approach may result in prohibitive running times as $\langle k \rangle$ increases.

Currently, there are no analytical results regarding the probability of self or multi-edges as well as bias for the subgraph connection process. To help develop some understanding, we set up the following experiment: using the CMA and the refusing algorithm we generate a series of homogeneous networks. Initially the CMA is parametrized with no $G_\triangle$ subgraphs and only G_0, reverting to the configuration model. For increasing degree of $k = 2, 3, 4, 5$ we then determine the average number of attempts required before a network is produced as well as the average number of $G_\triangle$ by-products. We then repeat this but with the CMA parametrized with an increasing number of $G_\triangle$ subgraphs, distributed so that a node is incident to at most one $G_\triangle$ subgraphs, and so on. Figure 2 illustrates that both the number of attempts and that of $G_\triangle$ by-products per network increase with degree, as one would expect. It also reveals that these quantities decrease when the CMA is parametrized with increasing numbers of $G_\triangle$ subgraphs, regardless of degree. Since the number of attempts per networks is a function of the number of self and

multi-edges, these results also imply that the number of self and multi-edges reduce as the number of $G_\triangle$ increases.

We believe the following to be an intuition behind this surprising result: consider a node incident to two $G_\triangle$ stubs. For a self-loop to be created about this node, there is a single opportunity: both hyperstubs must be simultaneously selected during the connection procedure. Now, if we consider a node with the same degree, but with each of its of stubs being used to form only lines, then there are $k(k-1)/2 = 6$ different ways in which pairs of stubs may be selected that result in a self edge. Thus, in general, hyperstubs will reduce the number of ways in which tuples of nodes may be connected, compared to stubs, and this will impact both the self and multi-edge probabilities.

Edge probability: With the subgraph connection process it is possible to replicate some of the estimates for the number of self and multi-edges that exist for the standard configuration model, as shown in [2]. The following calculations are intended to further develop intuition and by no means form a rigorous argument. Let us consider a network model composed of only $G_\triangle$ subgraphs, referred to henceforth as the $G_\triangle$ model. Let $3m_t$ denote the total number of $G_\triangle$ hyperstubs, i.e, this network has a total of m_t $G_\triangle$ subgraphs and $6m_t = 2m$ stubs, since each $G_\triangle$ hyperstub is composed of two stubs.

We first consider the probability of two nodes sharing a single edge in the $G_\triangle$ model. Let nodes i and j have $G_\triangle$ degrees of t_i and t_j, respectively. A single hyperstub of i may connect to any of the t_j hyperstubs originating from j. The probability of selecting one of j's hyperstubs is $t_j/(3m_t - 1)$, since we can no longer select the initial hyperstub incident to i. However, any one of i's hyperstubs could connect to any one of j's hyperstubs, and $t_i t_j/(3m_t - 1)$ correctly accounts for this. A third hyperstub must now be selected, incident to a third distinct node, i.e., any one of the $3m_t - t_i - t_j$ hyperstubs that are not incident to either i or j. If a hyperstub incident to i or j were to be selected this would result in both a self and multi-edge. Therefore, the probability of i and j sharing a single edge is given by

$$p_{i,j} = \frac{t_i t_j}{3m_t - 1}\left(\frac{3m_t - t_i - t_j}{3m_t - 2}\right) \tag{4}$$

Since the degree distribution is fixed and we are interested in the limit as $N \to \infty \Rightarrow m_t \to \infty$,

$$\lim_{N\to\infty} p_{i,j} = \frac{t_i t_j}{3m_t}. \tag{5}$$

Let us now consider that in the $G_\triangle$ model each node is incident to $2t_i = k_i$ stubs in a network composed of a total of $2(3m_t) = 2m$ stubs. By making these substitutions into equation (5) the edge probability of the $G_\triangle$ model can be compared to its equivalent configuration model

$$\frac{\frac{k_i}{2}\frac{k_j}{2}}{6m_t} = \frac{k_i k_j}{4m} < \frac{k_i k_j}{2m},$$

where the r.h.s. represents the edge probability in the configuration model. This counter-intuitive result for the $G_\triangle$ model is due to half of a node's stubs being obliged to connect to a third distinct node, excluding possibilities of i and j connecting which would otherwise be possible in the configuration model.

Multi-edge expectation: As in the standard configuration model, we can use equation (5) to estimate the number of multi-edges that may happen in two ways when selecting a triplet of nodes: (*a*) at least one of the constituent pairs already being connected or (*b*) all three of nodes already being connected. We first consider (*a*), the more likely scenario. i and j will share an edge with the probability given in

equation (5). To compute the probability of finding a second edge between nodes i and j in the $G_\triangle$ model, one must compound $(t_i - 1)(t_j - 1)/(3m_t - 1)$ with equation (5)

$$p(A(i,j) > 1) = \frac{t_i t_j (t_i - 1)(t_j - 1)}{(3m_t)^2},$$

summing this probability over all pairings of nodes and dividing by 2 to remove the double count, yielding

$$\begin{aligned} \lim_{N\to\infty} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} p(A(i,j) > 1) &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{t_i t_j (t_i - 1)(t_j - 1)}{(3m_t)^2} \\ &= \frac{1}{2(3m_t)^2} \sum_{j=1}^{N} t_j (t_j - 1) \sum_{i=1}^{N} t_i (t_i - 1) \\ &= \frac{1}{2} \left(\frac{\langle G_\triangle^2 \rangle - \langle G_\triangle \rangle}{\langle G_\triangle \rangle} \right)^2, \end{aligned} \tag{6}$$

where we have used

$$3m_t = \langle G_\triangle \rangle N, \quad \langle G_\triangle \rangle = \frac{1}{N} \sum_{i=1}^{N} t_i, \quad \langle G_\triangle^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} t_i^2. \tag{7}$$

We again compare this value to that of the standard configuration model with the substitutions $2t_i = k_i$ and $2(3m_t) = 2m$ yielding

$$\frac{1}{2} \left(\frac{\frac{1}{2}\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle} \right)^2 < \frac{1}{2} \left(\frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle} \right)^2,$$

where the r.h.s. represents Newman's original estimate for multi-edges in the configuration model [2]. Now we consider scenario (b), selecting the same triplet of nodes twice resulting in three multi-edges. Consider the nodes i, j and l with $G_\triangle$ degrees of t_i, t_j and t_l. This triple of nodes are connected with probability

$$\lim_{N\to\infty} p_{i,j,l} = \lim_{N\to\infty} \left(\frac{t_i t_j t_l}{3m_t(3m_t - 1)} \right) = \frac{t_i t_j t_l}{9m_t^2},$$

the probability of this triple being selected twice is approximately

$$\frac{t_i t_j t_l (t_i - 1)(t_j - 1)(t_l - 1)}{(3m_t)^4}.$$

This probability can be summed over all triplets of nodes yielding

$$\begin{aligned}
&\frac{1}{3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N}\frac{t_i t_j t_l (t_i-1)(t_j-1)(t_l-1)}{(3m_t)^4} \\
&\quad=\frac{1}{3(3m_t)^4}\sum_{i=1}^{N} t_i(t_i-1)\sum_{j=1}^{N} t_j(t_j-1)\sum_{l=1}^{N} t_l(t_l-1) \\
&\quad=\frac{1}{3N\langle G_\triangle\rangle}\left(\frac{\langle G_\triangle^2\rangle-\langle G_\triangle\rangle}{\langle G_\triangle\rangle}\right)^3, \qquad (8)
\end{aligned}$$

where we have again used equation (7). The expected number of multi-edges created by two $G_\triangle$ subgraphs connected on the same triplet of nodes is not constant with network size but instead tends to zero with increasing network size. This result, alongside equation (6), suggests that the number multi-edges in the $G_\triangle$ model will be less than what is found in the equivalent configuration model network. We next consider the probability of a self-edge in the $G_\triangle$ model.

The number of self-edges: During the connection process of the configuration model self-edges are created when two stubs that are incident to the same node are connected. The analogue of this in the $G_\triangle$ model is selecting three hyperstubs incident to the same node, resulting in three self-edges. We shall denote this event $\{i,i,i\}$

$$\begin{aligned}
p(\{i,i,i\}\wedge\{i,i,i\}) &= \frac{\binom{t_i}{3}}{(3m_t)(3m_t-1)}, \\
\lim_{N\to\infty} p(\{i,i,i\}\wedge\{i,i,i\}) &= \frac{t_i(t_i-1)(t_i-2)}{6(3m_t)^2},
\end{aligned}$$

this value can be summed over all nodes to estimate the expected number of self-edges in the network

$$\sum_{i=1}^{N}\frac{t_i(t_i-1)(t_i-2)}{6(3m_t)^2}=\frac{\langle G_\triangle^3\rangle-3\langle G_\triangle^2\rangle+2\langle G_\triangle\rangle}{6N\langle G_\triangle\rangle^2}, \qquad (9)$$

where we have used equation (7). This value, like equation (8) is not fixed with network size and instead tends to zero as N becomes large.

Node duplicates: In the $G_\triangle$ model it is possible to select a pair of hyperstubs incident to the same node alongside a distinct third node, resulting in a self- and multi-edge. We shall denote this event $\{i,i,j\}$. Then

$$\begin{aligned}
\lim_{N\to\infty} p(\{i,i,j\}) &= \lim_{N\to\infty}\left(\frac{\binom{t_i}{2}}{3m_t}\left(\frac{3m_t-t_i}{3m_t-2}\right)\right) \qquad (10) \\
&= \frac{t_i(t_i-1)}{2(3m_t)},
\end{aligned}$$

which, after summing over all nodes, yields

$$\sum_{i=1}^{N} \frac{t_i(t_i-1)}{3m_t} = \frac{\langle G_{\triangle}^2\rangle - \langle G_{\triangle}\rangle}{2\langle G_{\triangle}\rangle} \tag{11}$$

Since the determining factor of this expectation is the selection of a pair of hyperstubs incident to the same node we shall compare it to the self-edge probability for the equivalent configuration model network

$$\begin{aligned} \frac{\langle G_{\triangle}^2\rangle - \langle G_{\triangle}\rangle}{2\langle G_{\triangle}\rangle} &= \frac{\langle (k/2)^2\rangle - \langle k/2\rangle}{2\langle k/2\rangle} \\ &= \frac{\frac{1}{4}\langle k^2\rangle - \langle k\rangle}{\langle k\rangle} \\ &< \frac{\langle k^2\rangle - \langle k\rangle}{2\langle k\rangle}, \end{aligned} \tag{12}$$

i.e., as $N \to \infty$ we expect that the number of duplicate node selections resulting from $G_{\triangle}$ placement will be strictly less than the number of self-edges in the equivalent configuration model network.

By-products: It is possible for previously created subgraphs to become connected into a set of subgraphs with overlap, see Fig. 3 for an illustration. The expected number of multi-edges above demonstrates the possibility for one such occurrence, here, two $G_{\triangle}$ subgraphs sharing an edge. In this case if the multi-edge was collapsed down to a single edge, the process would yield a $G_{\boxslash}$ subgraph. The expected number of these events was shown to be bounded by a number of multi-edges in the equivalent configuration model network. However, this type of connection was not permitted in our implementation.

Currently, we are unable to offer estimates regarding the frequency of erroneous $G_{\triangle}$ subgraphs, that is, $G_{\triangle}$ subgraphs that appear beyond that which were controlled for. This type of connection is permitted in our implementation and would result in the subgraph by-products as shown in Figure 3. However, Fig. 2 indicates that the number of erroneous $G_{\triangle}$ subgraphs decreases as the number of intended subgraphs increases.

2.4 *The Big-V algorithm*

The Big-V algorithm does not generate networks as such, but is a widely-used, see [12, 27–29] for example, degree-preserving rewiring algorithm, making it possible to control clustering. At each iteration, the algorithm selects a linear chain of five nodes at random, e.g., $\{a, b, c, d, e\}$ with four edges $\{(a, b), (b, c), (c, d), (d, e)\}$. It then delete edges (a, b) and (d, e) to form (a, e) and (b, d). When starting from an unclustered network, this process will lead to at least one extra $G_{\triangle}$ being created [11]. This is repeated until the desired level of clustering is achieved. It is possible to include a Metropolis-style augmentation whereby at each step the local clustering coefficient is computed for the five nodes before and after rewiring, and the rewired configuration is only accepted if it results in an increase in average local clustering. It is worth noting that this algorithm leads to a positive degree–degree correlation that was not necessarily present in the original network.

In this article, we use the Big-V algorithm to demonstrate that our newly proposed algorithms are able to sample from a larger part of the state space of all possible networks with a given degree sequence and global clustering coefficient.

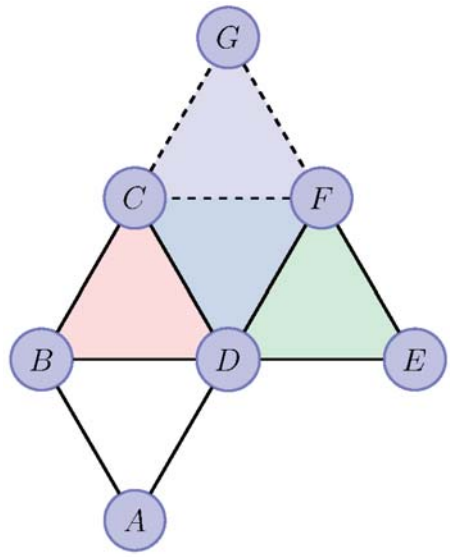

FIG. 3. Unintended generation of subgraphs with overlap. Despite satisfying the generation constraints given in Section 2.3, the addition of triangle (C,G,F) to toast (A,B,C,D) and triangle (D,F,E) results in three unintended distinct toasts {(B,C,F,D), (D,C,F,E) and (D,C,G,F)} overlapping on one unintended triangle (C,F,D).

2.5 *Models of contagion*

In order to illustrate the impact of network structure—and higher-order structure particularly—different epidemic dynamics were simulated on the generated networks. Three different models were chosen: susceptible-infected-susceptible (*SIS*), *SIR* and complex contagion [30, 31]. To simulate *SIS* and *SIR* dynamics, the fully susceptible network of nodes is perturbed by infecting a small number of nodes. Infected nodes spread the infection to susceptible neighbours at a per-link rate of infection τ. Infected individuals recover independently of the network at rate γ and become susceptible again (for *SIS* dynamics) or become removed (for *SIR* epidemics). In contrast to the infection process in the previous two dynamics, the complex contagion process requires that susceptible nodes are exposed to multiple infectious events before becoming infected. These events must be from different infectious neighbours as only the first infection attempt from an infectious node counts. This critical infection threshold for each node is set in advance and is usually bounded from above by the degree of the node. To simulate the complex contagion dynamics, nodes are allocated infection thresholds $r_i \in \mathbb{N}$, where $i = 1, 2 \ldots, N$, and the fully susceptible population of nodes is perturbed by infecting an initial number of nodes chosen at random. In this model a susceptible node i becomes infected as soon as it has received at least r_i infectious contacts from r_i distinct infectious neighbours. There is no recovery in this model and infected individuals remain infected for the duration of the epidemic.

3. Results

3.1 *Algorithm validation*

To validate our algorithms, we generated a number of networks with pre-specified degree distribution and subgraph set, as well as a multinomial distribution of subgraph corners or hyperstubs around nodes. We verified that the networks generated were as expected given the input.

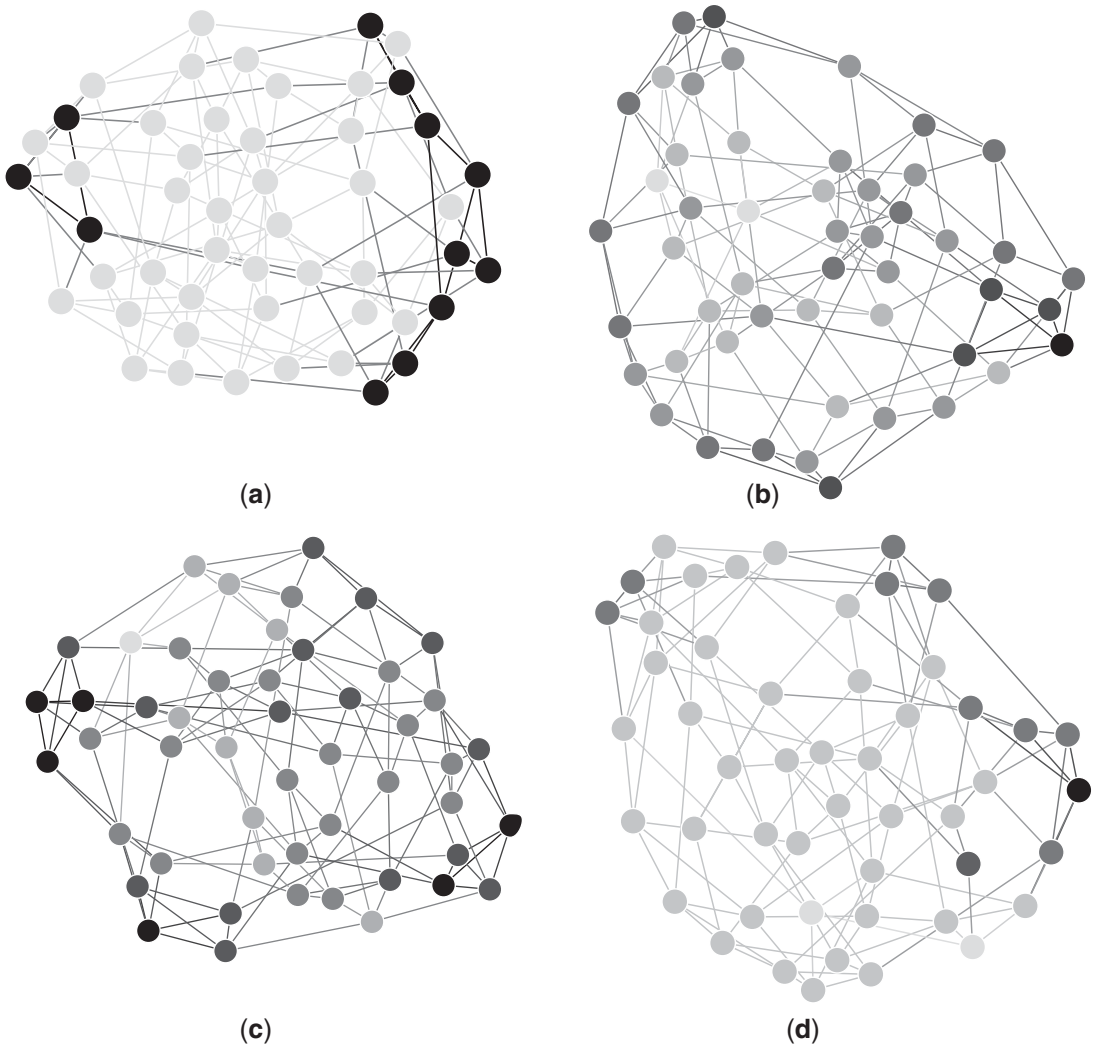

FIG. 4. Small networks generated by the Big-V, UDA and CMA algorithms. All networks have the same homogeneous degree sequence with $k = 5$. The Big-V algorithm rewired the random network, Fig. 4a. The UDA was parametrized with subgraphs G_0 $G_{\boxslash}$ and $G_{\boxtimes}$. The CMA was parametrised so that every node was incident to 2 $G_{\triangle}$. The Big-V, UDA, and CMA networks all have a global clustering coefficient of $C = 0.22$. The network nodes are coloured so that light/medium/dark grey denote nodes of low/medium/high clustering, respectively. (a) Random, (b) Big-V, $C = 0.22$, (c) UDA, $C = 0.22$ and (d) CMA, $C = 0.22$.

As described in Section 2, the algorithms preserve the degree sequence, permitting at most a single edge to be deleted if the degree sequence sums to an odd number. The ability to exercise control over the networks' subgraph topology is illustrated by Fig. 4. Note that Fig. 4a shows a *random* network that includes $G_{\triangle}$ subgraphs. When constructing networks using the configuration model it is possible to create $G_{\triangle}$ subgraphs with non-zero probability and this is to be expected [32]. However, this is a function of mean degree not network size, and this probability goes to zero with network size going to infinity.

TABLE 1. *Subgraph counts for the networks of Fig. 4. Note: if one adds a single $G_{\triangle}$ so that it shares a single edge with a $G_{\boxslash}$ and this edge is not the diagonal edge of $G_{\boxslash}$, then $d4$ increases by one but $t3$ will have only increased by one, not two. We note that $2 \cdot d3$ yields the maximum number of possible $G_{\triangle}$ induced by $G_{\boxslash}$. In general, calculating the number of $G_{\triangle}$ in this way will always yield the maximum possible count but not necessarily the true count because a single $G_{\triangle}$ could be shared by more than one $G_{\boxslash}$.*

	$c4$	$d4$	$e4$	$i4$	$s4$	$t3$	$u3$	$u4$
Random	0	0	42	17	446	6	482	1706
Big-V	1	23	10	10	212	7	386	1220
UDA	7	10	22	5	243	1	389	1239
CMA	0	9	10	40	185	24	389	1201

To properly demonstrate the proposed algorithms' control over the building blocks in the network, we used a recently described subgraph counting algorithm [12] to count the number of subgraphs *a posteriori*. In our implementation, we counted subgraphs composed of four nodes or less—see the top two rows of Fig. 1, as well as 5- and 6-cycles. Table 1 provides the subgraph counts for the networks displayed in Fig. 4. It confirms that the random network given in Fig. 4a contains 6 $G_{\triangle}$, counted uniquely, as observed above. The table also reveals that, through increasing the frequency of $G_{\triangle}$, the Big-V algorithm also introduced $G_{\boxslash}$ and $G_{\boxtimes}$ subgraphs. The UDA was parametrized with $\{G_0, G_{\boxslash}, G_{\boxtimes}\}$ and the table confirms a significant presence of these subgraphs when compared with the random network. Although the CMA was parametrized solely with $G_{\triangle}$ subgraphs distributed so that each node was incident to 2 $G_{\triangle}$ subgraphs, the subgraph counts reveal that this network contains 9 $G_{\boxslash}$ subgraphs. This is a consequence of attempting to generate *small* networks with such a high prevalence of triangles: it is highly likely that the algorithms will select nodes that already share one other common neighbour later in the connection process. One expects the proportion of these events to become increasingly negligible with greater network size.

Next, we used the above motif counting algorithm to evaluate the extent to which the proposed algorithms can exert control over the prevalence of subgraphs in the generated networks. Figure 5 compares *measured* counts of subgraphs in UDA and CMA networks with *expected* counts. Here, an important observation must be made at the outset. Even in random networks, cycles ($G_{\square}$, $G_{\pentagon}$ and $G_{\hexagon}$) appear in significant quantities: 33, 100 and 333 times, respectively, and regardless of network size. They are a natural consequence of the fact that the probability of selecting two nodes in different branches of a finite tree-like network is non-zero. Therefore, our *expected* counts are the sum of the counts expected *by construction* and those *measured* in the random networks. For example, since the CMA networks were generated with each node being incident to a single $G_{\hexagon}$ subgraph, a total of 833 uniquely counted $G_{\hexagon}$ subgraphs were expected *by construction* in networks of size $N = 5000$. However, because an average of 344 $G_{\hexagon}$ subgraphs were counted in random networks of size $N = 5000$, our *expected* count was $833 + 344 = 1177$. The *measured* count was found to be 1165. More generally, we found the *expected* counts to match well with the *measured* counts, indicating that the generating algorithms did not create by-products in addition to those observed at random.[1]

[1] Although we will show in Section 3.3 that for specific parameterizations of CMA, by-products are possible.

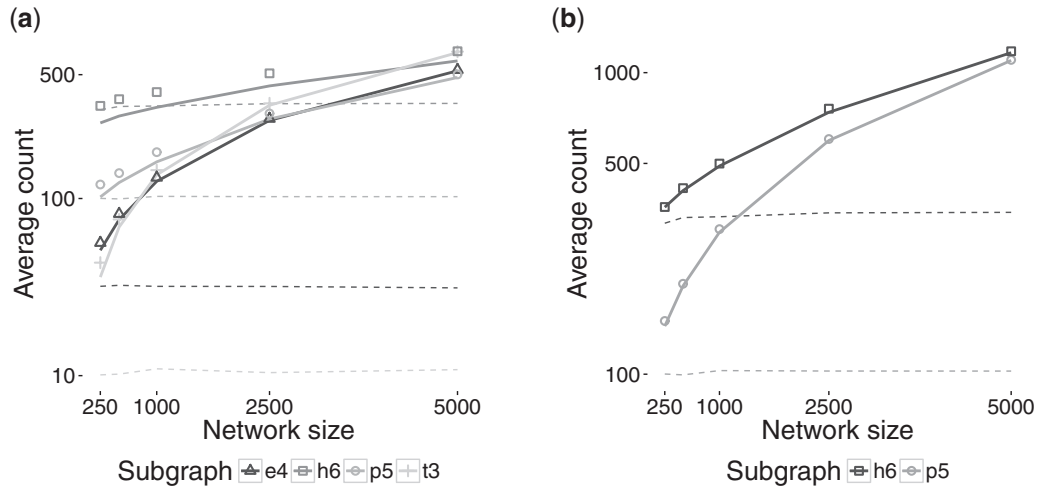

FIG. 5. A comparison of subgraphs found in the UDA and CMA networks to their random network analogues and expected counts plotted with thick lines, thin lines and discrete markers, respectively. $p5$ and $h6$ denote the counts of $G_{\pentagon}$ and $G_{\hexagon}$, respectively. All networks have the same homogeneous degree sequence with $k = 5$ but with increasing size: $N = 250, 500, 1000, 2500, 5000$, where 100 of each size was generated. (a) The UDA algorithm was parametrized with subgraphs $\{G_{\triangle}, G_{\square}, G_{\pentagon}, G_{\hexagon}\}$, and the resulting average subgraph counts are shown on the left. (b) The CMA algorithm was parametrized so that each node was incident to a single $G_{\pentagon}$ and $G_{\hexagon}$ subgraph, and the resulting average subgraph counts are shown on the right. The expected values were calculated by summing the total counts from the subgraph sequences, dividing them by the subgraphs' node cardinality, and adding these figures to the number of subgraphs found as by-products in the random networks.

However, these results also suggest that the level of control exerted by the algorithms over subgraph prevalence depends on how often those subgraphs appear naturally as by-products. Control is strongest for subgraphs that do not appear naturally as by-products. When considering subgraphs that appear naturally with high frequency, e.g., $G_{\pentagon}$, real control over their prevalence can only be achieved if an even higher frequency is imposed, which may not always be possible for a given degree sequence and global clustering.

In what follows, we set out to highlight differences between the new algorithms compared to classic ones and also to emphasize the diversity within networks generated by the same algorithms.

3.2 *Sampling from a different area of the network state space*

In this section, we seek to highlight the versatility of the proposed generation mechanisms by showing that, given a degree distribution and a global clustering, they sample different areas of the network state space than existing methods such as Big-V. We begin by reminding the reader that the Big-V algorithm searches for paths of five nodes and rewires such paths so that additional triangles are created. In other words, the principal building block of this algorithm is the $G_{\triangle}$ subgraph and subgraphs that may be constructed by overlapping $G_{\triangle}$ subgraphs. It follows that this algorithm is unlikely to give rise to a higher than expected at random number of $G_{\square}$ or other 'empty' cycles. The UDA was therefore parametrized with subgraph family $\{G_0, G_{\triangle}, G_{\square}, G_{\pentagon}, G_{\hexagon}\}$. In order to eliminate the effect of degree heterogeneity, a homogeneous degree sequence with $k = 5$ was used. The resulting networks had a global clustering coefficient of $C = 0.04$, induced by 666 (uniquely counted) $G_{\triangle}$ subgraphs. We then used the Big-V algorithm to rewire random networks constructed using the same degree sequence until the desired level of clustering,

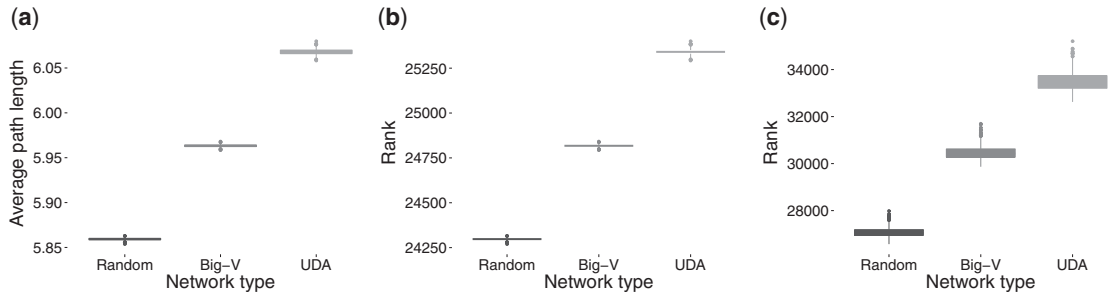

FIG. 6. Plots of the average path length and diameter for homogeneous networks ($N = 5000$ and $k = 5$) for network family **A**. The Big-V algorithm was parametrized solely by clustering, in this case $C = 0.04$, to best suit the networks produced by the UDA. The differences in average path length, average betweenness centrality and maximum betweenness centrality between the random network and its Big-V analogue were of similar magnitude as the differences between the Big-V network and the cycle-based UDA networks, and these were significant. (a) Average path length, (b) average betweenness and (c) maximum betweenness.

$C = 0.04$, was achieved. Significant differences between generated networks would confirm that the Big-V and UDA generated networks are sampled from different areas of the state space of networks satisfying that degree sequence and global clustering. As a further point of reference, data taken from a random network realization, generated using the configuration model, of the degree sequence were included in all of our analyses. Henceforth we shall refer to these three types of networks as network family **A**.

In Fig. 6, the distributions of the average path length, average betweenness centrality and maximum betweenness centrality for the above networks are given. In general, an increase in clustering results in a higher value of the average path length—see the average path length of random and Big-V networks in Fig. 6a. This is a known result [11]. Surprisingly, a similar magnitude of difference in average path length and average and maximum betweenness centrality is observed between the Big-V and UDA networks despite them having the same global clustering, see Figure 6(a–c), respectively. Output from the subgraph counting algorithm (Fig. 7) confirms that, as expected, the Big-V algorithm does not generate more $G_{\square}$ subgraphs than are observed in the random network. More generally, the results show that the Big-V and UDA networks exhibit markedly different subgraph topologies with the Big-V networks relying heavily on $G_{\boxslash}$ to cluster the networks unlike UDA networks that rely almost exclusively on $G_{\triangle}$ not appearing as part of any other subgraph. It may be that such variation was facilitated by the low level of clustering considered, and that with higher clustering, eliciting such differences might be more challenging. However, these results provide evidence that the UDA can sample from a different part of the state space than the Big-V algorithm.

3.3 *Diversity within the newly proposed algorithms*

In this section, we illustrate the diversity of networks generated with UDA and CMA by exploring the impact of subgraph distribution over nodes (for identical degree distribution and global clustering) and how it may change network characteristics.

To do this we first parametrized the UDA with subgraph family $\{G_0, G_{\triangle}, G_{\square}, G_{\boxslash}, G_{\boxtimes}\}$ (chosen due to its frequent use in the literature, e.g., [11–13, 21, 33, 34]), and a heterogeneous degree sequence generated using the Poisson distribution with $\lambda = 5$. Since it is difficult to control the number of subgraphs that appear in a network generated using the UDA, we counted the total number of each subgraph, from UDA-produced subgraph sequences, and used these counts to create alternative subgraph sequences as

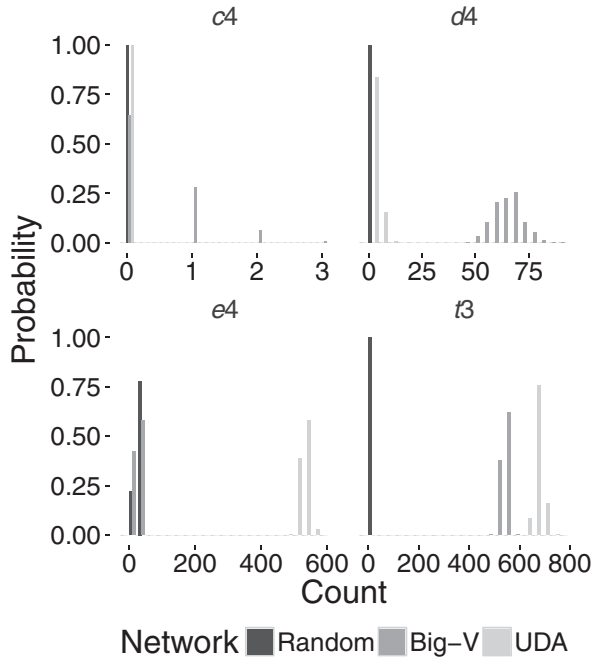

FIG. 7. Distributions of total number of subgraphs in network family **A** ($N = 5000$, $k = 5$). The Big-V and UDA networks have a global clustering coefficient of $C = 0.04$. All given counts are unique. The $t3$ counts denote the number of $G_{\triangle}$ subgraphs that are not involved in any subgraphs of four nodes (i.e., $G_{\boxslash}$ and $G_{\boxtimes}$). However, the $c4$ and $d4$ counts may include $G_{\triangle}$ subgraphs shared by $G_{\boxslash}$ and $G_{\boxtimes}$. The number of $G_{\square}$ subgraphs generated by the Big-V algorithm is very close to the counts found in random networks.

input to the CMA, see Section 2.2, rather than drawing such sequences from a theoretical distribution. The resulting networks were therefore expected to have identical degree sequence, global clustering of 0.13 and subgraph counts. Since the CMA allows us to choose arbitrary sequences of subgraphs, we opted to push the clustered subgraphs, $\{G_{\triangle}, G_{\boxslash}, G_{\boxtimes}\}$, onto the higher-degree nodes to accentuate the effect of clustering. We did this by specifying that these subgraphs had to appear with multiplicity greater than one. For example, a degree-three $G_{\boxtimes}$ hyperstub required a minimum $k = 9$-degree node. As previously, we included a random network realization of the heterogeneous degree sequence for comparison. Henceforth, we shall refer to these three types of networks as network family **B**.

The heterogeneity in degree distribution allows us to use additional degree–dependent metrics: degree–degree correlations and degree–dependent clustering [10, 17]. These have been plotted in Fig. 8. The plot for the degree–degree correlation coefficient shows that by aggregating clustered subgraphs around high-degree nodes, the CMA-constructed networks yield a higher assortativity than that of UDA and random networks, see Fig. 8a. This is an important property of the methodology since the clustering potential of a network is bounded by the degree–degree correlation coefficient [10]. Moreover, if one wishes to maximize clustering in heterogeneous networks, it is necessary for nodes of similar degree to mix preferentially. Figure 8b shows that the CMA networks yield a negatively skewed distribution of degree-dependent clustering, with nodes of degree $k \geq 9$ contributing most to clustering. The ability to manipulate the degree and clustering relationship as well as assortativity clearly demonstrates the

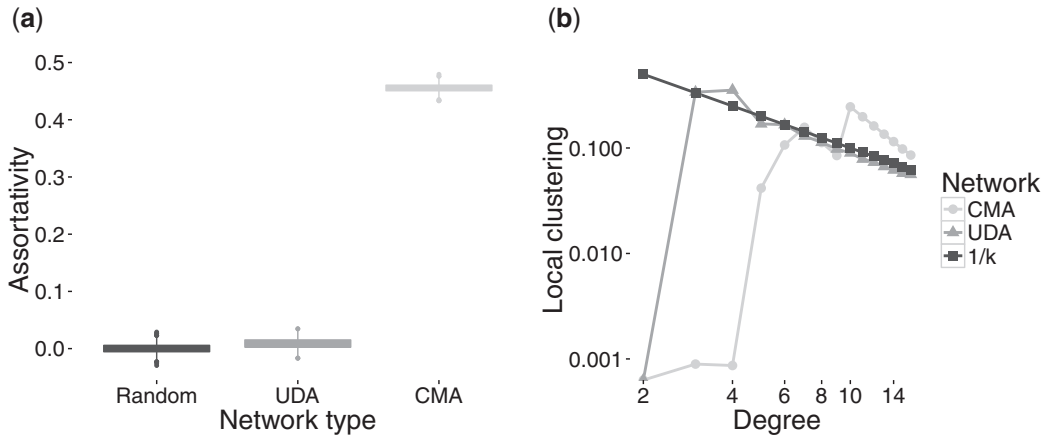

FIG. 8. Plots of assortativity and degree-dependent average local clustering for network family **B** with $k \sim \text{Pois}(5)$. The UDA and CMA networks have a global clustering coefficient of $C = 0.13$. The distribution of subgraphs in CMA networks was manipulated so that the clustered subgraphs $\{G_{\triangle}, G_{\square}, G_{\boxtimes}\}$ appeared around nodes with multiplicity greater than one. In order to preserve the subgraph degree sequence, these aggregated subgraphs were allocated to the higher degree nodes, resulting in higher assortativity and a more positively skewed distribution of degree-dependent clustering. (a) Assortativity and (b) degree-dependent clustering

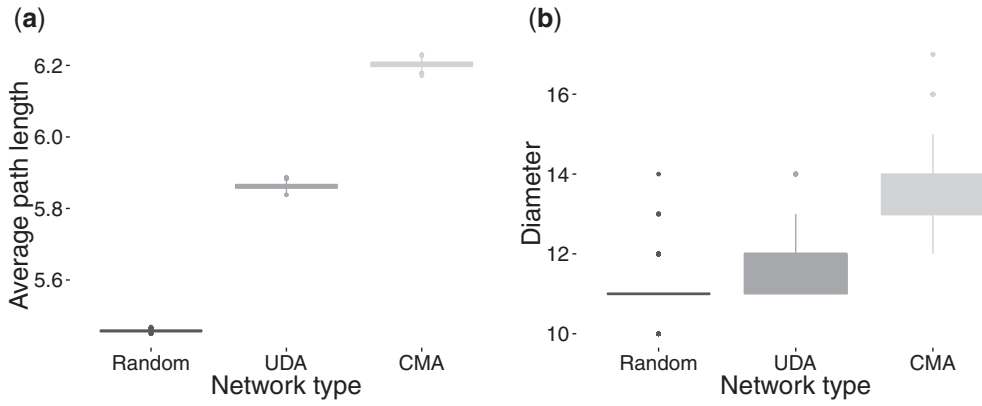

FIG. 9. Plots of average path length and diameter for network family **B** with $k \sim \text{Pois}(5)$. The UDA and CMA networks have a global clustering coefficient of $C = 0.13$. The increased average path length and diameter between the UDA and random networks is attributable to the higher clustering. The similar increase between UDA and CMA networks is a reflection of the higher assortativity of the CMA networks. (a) Average path length and (b) diameter.

broader scope of the CMA when sampling from the ensemble of networks with same degree distribution and global clustering.

As with network family **A**, an increase in average path length, diameter, average and maximum betweenness centrality of UDA and CMA networks over random networks will be attributable to the increased global clustering coefficient, $C = 0.13$, see Figs. 9 and 10. However, since UDA and CMA networks share the same degree sequence and global clustering coefficient differences in these metrics between UDA and CMA can only be due to increased degree–degree correlation and negatively skewed

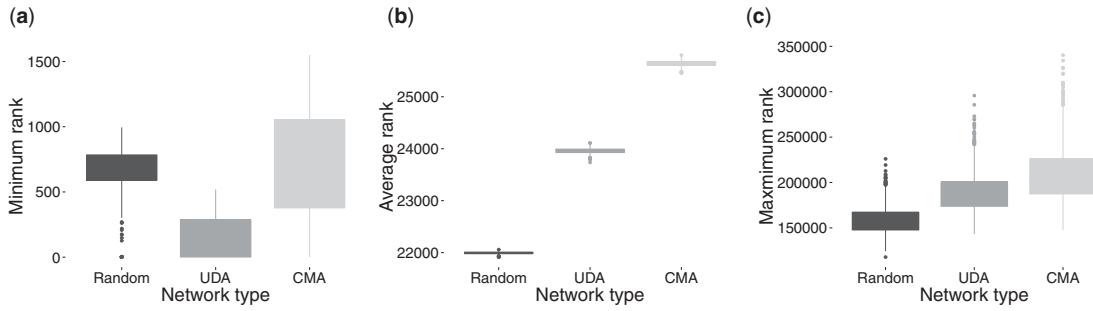

FIG. 10. Plots of betweenness centrality for network family **B** with $k \sim \text{Pois}(5)$. The UDA and CMA networks have a global clustering coefficient of $C = 0.13$. A trend of increasing average and maximum betweenness centrality is observed between random, UDA and CMA networks, respectively. (a) Minimum, (b) average and (c) maximum.

distribution of degree-dependent clustering. It has previously been noted that increased assortativity corresponds to an increase in average path length [35], and this will be compounded by the higher-degree nodes (which inevitably serve as central hubs) being more clustered. Similarly, an increase in diameter (a function of path length) will be due to these highly clustered high-degree nodes. Finally, Figs. 10b and c show a significant increase in average and maximum betweenness centrality between UDA and CMA networks. This is yet another manifestation of the presence of these highly clustered high-degree nodes.

Table 2 presents a comparison between *measured* and *expected* average subgraph counts for the networks in family **B**. Although there is good agreement for UDA networks, it is observed that CMA networks have produced by-products other than what was expected at random, e.g., an additional 50% $G_{\boxslash}$ have appeared as by-products. The effects of finite size have been exacerbated by aggregating clustered subgraphs around higher degree nodes, effectively excluding lower to medium degree nodes during this part of the connection process. Within this densely connected component, it is easy to envisage a situation where adding only a single edge may create additional (unwanted) subgraphs. This highlights the fact that whilst the total number of $G_{\triangle}$ is preserved (as evidenced by identical global clustering), the way these subgraphs contribute to higher-order structure can vary significantly.

This section has highlighted that control over the choice of subgraph families and their distributions makes it possible to flexibly explore the solution space of networks with the same degree distribution and global clustering. This in turn provides us with the means to investigate specific areas of this solution space as well as further our understanding of how network metrics deal with such diversity.

3.4 *Does higher-order structure matter?*

In order to answer this question we make use of the network families **A** and **B** detailed above and test the impact of higher-order structure by considering the outcome and evolution of widely used dynamics on networks, namely, *SIS*, *SIR* and the complex contagion model.

For each network type in families **A** and **B** a series of networks, were generated. For each network, we performed a single Gillespie realisation of the *SIS*, *SIR* and complex contagion epidemics. The mean time evolution of infectious prevalence was then calculated, plotted and compared between network types. Complex contagion dynamics was simulated in a similar way but without recovery and remembering that a single infectious contact was usually not sufficient to result in an infected node. Different thresholds of infection and infectious seeds were used and these are specified in figure captions. Matlab code for the

TABLE 2. *Subgraph counts for network* $\boldsymbol{B}$ *(*$N = 5000$*,* $k \sim Pois(5)$ *and* $C = 0.13$*). The counts are unique. The expected counts are computed by summing the total counts from the subgraph sequences, dividing them by the subgraphs' node cardinality, and adding these figures to the number of subgraphs found as by-products in the random network. The counts for* $t3$ *are for* $G_{\triangle}$ *subgraphs that do not appear in any other subgraphs*

	$c4$	$d4$	$e4$	$t3$
Random	0	0	79	21
UDA	243	504	587	718
CMA	232	743	772	691
Expected	243	504	619	741

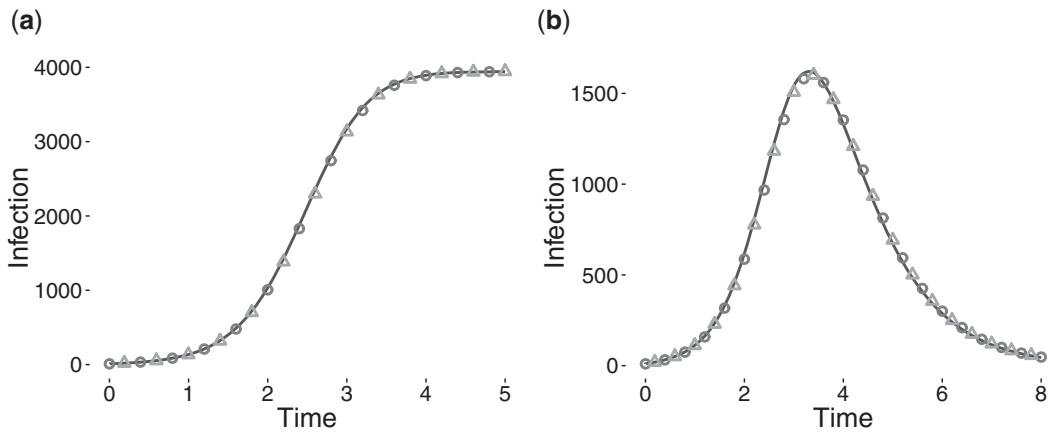

FIG. 11. (a) SIS and (b) SIR epidemic dynamics for network family **A**. The random, Big-V and UDA data have been plotted with a solid line, circle and triangle markers, respectively. The *SIS* and *SIR* epidemics represent the average of single Gillespie simulations on each of the 1000 network realizations from each network generation algorithm. The *SIS* and *SIR* epidemics were seeded with an initial infectious seed of $I_0 = 10$ and had a per link rate of infection of $\tau = 1$ and recovered independently at rate $\gamma = 1$.

SIS and *SIR* Gillespie algorithms is available from https://github.com/martinritchie/Dynamics. Accessed on 23 April 2016.

We know by construction that members of network family **A** were generated using different subgraphs, and Section 3.3 has shown that observable differences were found between networks in terms of average path length, betweenness centrality and subgraph composition. Despite this, Fig. 11, which show the time evolution for *SIS* and *SIR* dynamics, respectively, illustrate that these dynamics can display a certain degree of insensitivity to these differences in structure. In this case, it is the *SIR* dynamics that show the greatest difference, in peak infectious prevalence (Fig. 11b) albeit quite marginal.

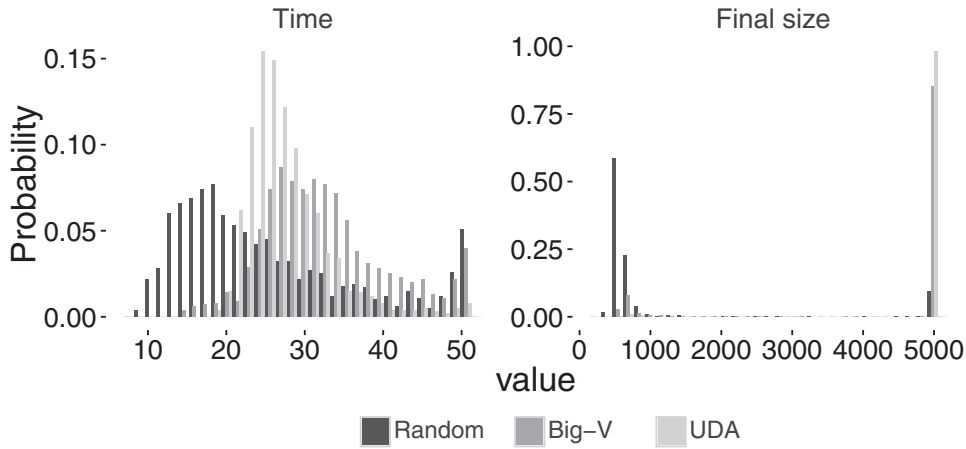

FIG. 12. Complex contagion dynamics for network family **A**. The complex contagion epidemics we parametrized an initial infectious seed of $I_0 = 250$ and a fixed threshold of infection of $r = 2$.

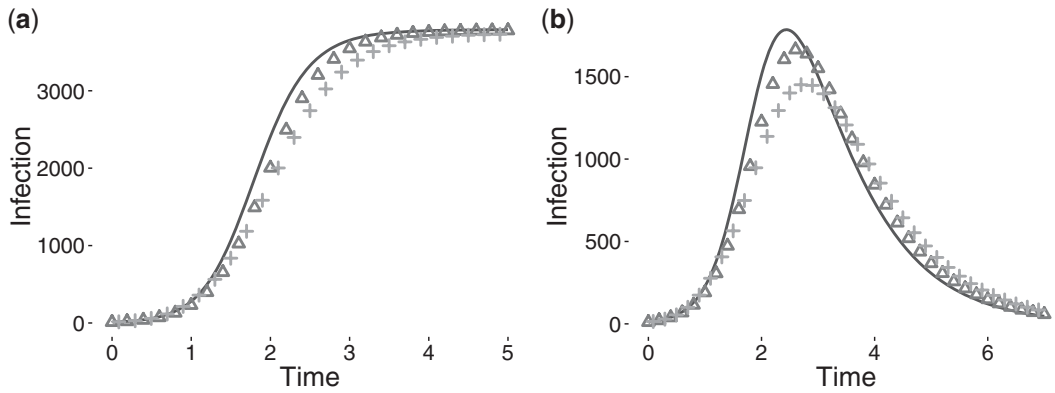

FIG. 13. (a) SIS and (b) SIR epidemic dynamics for network family **B**. The random, UDA and CMA data have been plotted with a solid line, triangle and cross markers, respectively. The *SIS* and *SIR* epidemics represent the average of single Gillespie simulations on each of the 1000 network realizations from each network generation algorithm. The *SIS* and *SIR* epidemics were seeded with an initial infectious seed of $I_0 = 10$ and had a per link rate of infection of $\tau = 1$ and recovered independently at rate $\gamma = 1$.

In contrast, complex contagion dynamics do show sensitivity to structural differences found between Big-V and UDA networks. Figure 12 reveals that for UDA networks the epidemic fully percolates in almost 100% of the simulations instead of only 80% of the cases for Big-V networks and that epidemics on UDA networks achieve this steady state in less time. This indicates that whilst UDA networks operate in the super critical regime, Big-V networks are closer to the transition point. Locating this transition is possible but is beyond the scope of this article.

When network family **A** is used, the networks' degree distribution and clustering appear to be the main determinants of the time evolution and outcome of the *SIS* and *SIR* epidemics. In contrast, when network family **B** is used, Figs. 13 and 14 show that all dynamics considered are impacted by differences in network

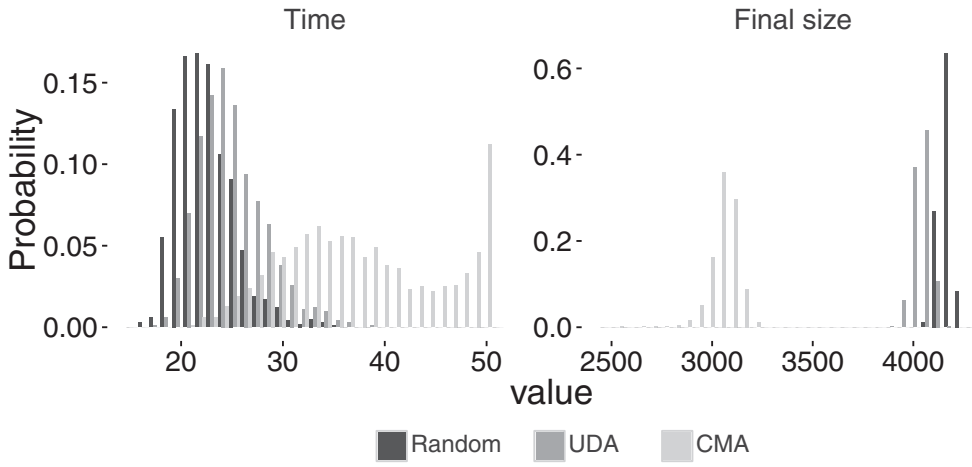

FIG. 14. Complex contagion dynamics for network family **B**. The complex contagion epidemics had an initial infectious seed of $I_0 = 1000$ and a fixed threshold of infection of $r = 3$.

topology. For Figs. 13a and b, a trend of inhibited spread of infection is observed from the random to UDA to CMA networks. It has already been shown that clustering slows the spread of infection [29, 36], and we see that this effect dominates over higher assortativity, which usually leads to faster initial spread of the epidemic [37]. Similarly, Fig. 14, which shows the distribution of the final epidemic size for the complex contagion dynamics, reveals that (a) the higher clustering observed in the UDA networks fails to have a significant impact when compared with the random network equivalent and (b) the CMA networks significantly slow the pace of the epidemic as well as reduce its final size compared with both random and UDA networks. Hence, for the UDA and CMA networks where both degree distribution and global clustering are identical the observed differences are explained by the combined effect of varying distributions of subgraph around nodes and varying prevalence of subgraphs (both of which are related to one another to some extent) as shown by Table 2.

Taken together, our simulation data show that even though the proposed algorithms construct networks with identical degree sequence and global clustering, these networks can give rise to measurable differences in resulting epidemics, be it in time evolution or final outcome. With the exception of *SIS* and *SIR* epidemics on network family **A** (still with some small differences), we found significant differences in all other instances. A more systematic investigation of more network models and wider parameter range for the dynamics is needed but left to future work.

4. Discussion

In this article, we have described two novel network generating algorithms that strictly preserve a given degree sequence whilst permitting control over the building blocks of the network and enabling the tuning of global clustering. We have compared these algorithms to one another as well as to the widely used Big-V rewiring algorithm. Using our algorithms we have empirically demonstrated that it is possible to create networks that are identical with respect to degree sequence and global clustering, yet elicit significant differences in network metrics and in the outcome of dynamical processes unfolding on them. We have presented evidence to suggest that the methods sample from different areas of the network state space and that these sampling variations do matter.

Of the two algorithms proposed, UDA is the simplest to use. We believe that this algorithm, when parametrized with complete subgraphs, would be more likely to yield analytical results. Note that whilst varying levels of clustering can be achieved and estimated before network construction it is not possible to target a specific level of clustering due to the emergent nature of the distribution of subgraphs around nodes. When constructing networks with incomplete subgraphs the UDA must decompose a certain number of hyperstubs back into stubs when generating sequences associated with incomplete subgraphs, which will introduce some bias. However, this source of bias is removed when using only complete subgraphs.

The CMA algorithm is more complex but also more versatile. Being able to build networks based on prespecified distributions of subgraphs alongside a given degree sequence, and preserve both, is highly novel. However this algorithm also contains a source of bias. In this case, when finding a node to accommodate a certain number of hyperstubs, the algorithm only considers nodes of suitably high degree. We conjecture that the algorithm should consider all nodes, regardless of degree, uniformly and if an invalid selection is made the algorithm should restart the process anew. However, this may result in prohibitive running times, especially when the total number of stubs contained within hyperstubs is close to the total number of stubs specified by the degree sequence.

The proposed connection procedure for subgraph based networks has revealed some surprising results. We found evidence that the number of self, multi-edges and erroneous $G_{\triangle}$ subgraphs in these networks is less than what is found in the equivalent configuration model networks. We should also point out that the connection procedure is not without its source of bias, namely, our reliance on the repetition procedure, whereby if a hyperstub selection results in self or multi-edges we return the hyperstubs to their respective bins and make a new selection. The alternative would be the refusal method, whereby an incompatible selection of hyperstubs requires the whole process to start anew. However, such approach leads to prohibitive running times. Note that the bias of the implemented connection process method may be offset by the overall reduction in self and multi-edges when connecting subgraphs. However, all these points ideally warrant supporting analytical results or, at least, further computational evidence.

The proposed models are unique and although the methods we implemented do suffer from biases, they were critical to being able to generate the desired networks. Importantly, there is currently no way to assess or measure the extent of these biases. This is because for a given degree distribution and given global clustering coefficient, there is currently no ground truth model nor is the entire state-space of such networks known in full. In light of this, we have taken the pragmatic step to focus on characterizing the network structure in terms of diversity. Being able to obtain diversity within networks sampled from the same part of the state space of such networks will be a critical component of constructing suitable null models.

In this respect, we have shown that significant diversity in networks with identical degree distribution and global clustering can be elicited. This has occurred in two ways: (1) by construction, i.e. changing subgraph families or redistributing the same number of subgraphs and (2) unexpectedly, through the emergence of by-products. We conjecture that any controlled—or believed to be controlled—network generation algorithm will yield by-products, unless heuristic constraints are introduced to reduce the likelihood of subgraphs sharing lower-order subgraph components for example. As witnessed in our results, even configuration model networks lead to a large number of loops with 4–6 nodes (longer cycles were not measured). This problem can only be exacerbated when control of more sophisticated structures is implemented. As such, care has to be taken when parametrizing algorithms. For example, one would need to specify a relatively large number $G_{\bigcirc}$ subgraphs in a network's construction to impact the subgraph count beyond what one would observe by chance in a random network. More surprisingly, as we witnessed with $G_{\boxslash}$ subgraphs in the CMA networks from network family **B**, significant numbers

of subgraph by-products can appear in addition to what was observed in the random networks depending on how one wishes to place the subgraphs around nodes.

We have seen that by using a modest selection of subgraphs, we have been able to substantially influence dynamics running on the network, particularly complex contagion dynamics. All results relating to this model indicate that constraining a network by degree sequence and clustering is not sufficient to accurately predict the course of the epidemic. More importantly, the results appear to suggest that the location of the critical regime depends on the higher-order structure of the network (above and beyond clustering).

Being able to generate networks with different structural properties or higher-order structure is a key feature of any network construction algorithm. However, if such structural details do not impact on dynamics unfolding on the network, then models for such dynamics can rely with high confidence on a limited set of network descriptors. Although degree sequence, degree–degree correlations and global clustering coefficient were observed to be the main drivers of disease transmission in models such as *SIS* and *SIR*, we found it not to be true in general. This is an important finding because one should remember that the dynamics simulated here are modest in complexity, when compared with models of neuronal dynamics for example, and yet, we were able to elicit significant differences by simply tuning the network structure above and beyond triangles. This implies that determining the role and impact of higher-order structure may yet hold and reveal many important and surprising results.

Acknowledgements

MR gratefully acknowledges Engineering and Physical Sciences Research Council (EPSRC, Doctoral Training Grant EP/K503198/1) and the University of Sussex for funding for his PhD. We would also like to thank Dr J.C. Miller for useful discussions on the complex contagion model [30], and for sharing his code for simulating the complex contagion model on networks [38].

Appendix

A.1 *Integer partitions*

The set of partitions of a positive integer, k, lists all possible ways of writing k as the sum of other positive integers. For example, the partition space of 3 is: $\{\{3\},\ \{2,1\},\ \{1,1,1\}\}$. The number of ways to partition an integer is given by the *partition function*. For this derivation only we use $p(k)$ to denote the partition function evaluated at k, i.e., the number of partitions of the integer k. In the case above, $p(3) = 3$. For $k = 1, 2, 3, 4, 5, 6\ldots$ the partition function returns $p(k) = 1, 2, 3, 5, 7, 11, \ldots$, respectively, and by convention $p(0) = 1$ and $p(-k) = 0$. This function can be used to calculate the number of times an integer $\alpha < k$ appears in the partitions of k. We first compute the number of partitions in which α will appear at least once. To illustrate the process we write k as a partition in the following way $\{k - \alpha, \alpha\}$ and use this initial partition as a starting point to list all remaining partitions of $k - \alpha$, for example:

$$\begin{aligned}
&\{k - (\alpha + 0), \alpha\}, \\
&\{k - (\alpha + 1), \alpha, 1\}, \\
&\{k - (\alpha + 2), \alpha, 2\},\ \{k - (\alpha + 2), \alpha, 1, 1\}, \\
&\{k - (\alpha + 3), \alpha, 3\},\ \{k - (\alpha + 3), \alpha, 2, 1\},\ \{k - (\alpha + 3), \alpha, 1, 1, 1\}, \\
&\ldots,
\end{aligned} \tag{A.1}$$

i.e., there will be $p(k-\alpha)$ such partitions. To more formally show this we use Euler's partition theorem

$$\sum_{n=0}^{\infty} p(n)x^n = \prod_{k=0}^{\infty} \frac{1}{1-x^k}. \tag{A.2}$$

To calculate values of p, we first expand the r.h.s of the above

$$(1+x+x^2+\dots)(1+x^2+x^4+\dots)(1+x^3+x^6+\dots)\cdot\dots,$$

such that to find the value of, e.g., $p(2)$, we simply collect the powers of x^2 to reveal the coefficient

$$p(2)x^2 = 2x^2.$$

In general, the terms in the geometric series in powers of k, i.e., $(1+(x^k)^1+(x^k)^2+\dots)$, give the number of times the integer k may contribute to n for the x^n term, assuming that $k \leq n$. For example, x^5 can be formed by $(x^2)^2(x^1)^1$, which means that $5 = 2+2+1$ or by $(x^1)^3(x^2)^1$ which means that $5 = 1+1+1+2$, or $(x^3)^1(x^2)^1$ which means that $5 = 3+2$. To prove that $p(k-\alpha)$ gives the number of partitions which α appears at least once, we write the following

$$(1+x+x^2+\dots)(1+x^2+x^4+\dots)\cdot\dots\cdot(x^\alpha+x^{2\alpha}+\dots)\cdot\dots,$$

where the exclusion of the $x^{0\alpha}$ term guarantees that the power of each and every term includes at least one *alpha*. Then we rewrite the terms to the power of α as

$$\begin{aligned} x^\alpha + x^{2\alpha} + \dots &= \frac{1}{1-x^\alpha} - 1 \\ &= \frac{x^\alpha}{1-x^\alpha}. \end{aligned} \tag{A.3}$$

Euler's theorem can be modified to account for such a term

$$x^\alpha \sum_{n=0}^{\infty} p(n)x^n = x^\alpha \prod_{k=0}^{\infty} \frac{1}{1-x^k}. \tag{A.4}$$

Comparing like-for-like powers in this modified expression gives the coefficient of x^n as $p(n-\alpha)$. Similarly, by writing

$$x^{m\alpha}(1+x^\alpha+x^{2\alpha}+\dots+x^{(m-1)\alpha}+x^{(m+1)\alpha}+\dots),$$

the result holds for multiples of α: $p(n-m\alpha)$, the number of partitions in which α appears at least m times. Using the cumulative property of this expression, it is possible to compute the number of partitions in which α appears exactly m times

$$p(k-\alpha(m-1)) - p(k-\alpha m)$$

multiplying this by m and summing over all multiples of α, $m : m\alpha \leq k$ will give the number of times that α appears in the partitions of k

$$p(k,\alpha) = \sum_{m=1}^{\lfloor \frac{k}{\alpha} \rfloor} m\big[p(k-\alpha m)) - p(k-\alpha(m+1))\big].$$

A.2 *Pseudocode for UDA*

Algorithm 1: Pseudocode for UDA. This pseudocode focuses on the salient points of the UDA, namely, how the algorithm draws solutions from the solution space of an underdetermined Diophantine equation to determine the arrangement of hyperstubs around a particular node. Other steps, such as ensuring the handshake lemma is satisfied for both lines and subgraphs, are detailed in Section 2.1 and can be viewed in the source code. The output hyperstub degree sequence H must be used as input for a modified configuration model connection process to realize a network, see Section 2.3.

input : $D = (d_1, d_2, \ldots, d_N)$, $G = \{G_1, G_2, \ldots, G_l\}$
output: $H \in \mathbb{N}_0^{l \times N}$.
Variables
D: degree sequence, N: number of nodes,
G: set of subgraphs, l: number of subgraphs,
g_i: subgraph adjacency matrix, X_k: solution space for degree k,
H: hyperstub degree sequence
Procedure
for *Each subgraph,* G_i **do**
    % Identify the degree sequences of the subgraphs.
    $s_i = \sum g_i$
    % Take the unique elements.
    $s_i = unique(s_i)$
end
% Concatenate into a single vector.
$S = (s_1, s_2, \ldots s_l)$
for $k = 1, 2, \ldots k_{\max}$ **do**
    % $X_k(i,:)$ denotes a hyperstub arrangement for a degree k node.
    $X_k = diorecur(S, k)$
end
for n = *1,2,...,* N **do**
    % Take random element from the solution space.
    $r = rand$; $h_n = X_{D(n)}(r, \cdot)$
end
% Concatenate into a single matrix.
$H = (h_1, h_2, \ldots, h_l)$
return

A.3 *Pseudocode for CMA*

Algorithm 2: Pseudocode for CMA. Other steps, such as ensuring the handshake lemma is satisfied for both lines and subgraphs, are identical to what is used for the UDA and are detailed in Section 2.1 and can be viewed in the Matlab source code. The output hyperstub degree sequence H must be used as input for a modified configuration model connection process to realize a network, see Section 2.3.

input : $D = (d_1, d_2, \ldots, d_N)$, $G = \{G_1, G_2, \ldots, G_l\}$, $S = \{S_1, S_2, \ldots, S_l\}$.
output: $H \in \mathbb{N}_0^{|s|\times N}$.
Variables
D: degree sequence, N: number of nodes,
G: set of subgraphs, l: number of subgraphs,
S: subgraph sequence, g_i: subgraph adjacency matrix,
$|s|$: number of unique corners in a subgraph, H: hyperstub degree sequence
Procedure
for *Each subgraph,* G_i **do**
    % Identify the degree sequence, s, of the subgraph.
    $s_i = \sum g_i$, $s_i = unique(s_i)$, $m = length(s_i)$
    % p reflects the proportions of hyperstubs
    $p_i = (p_1, p_2, \ldots, p_m)$
    for $j = 1, 2, \ldots, N$ **do**
        % The subgraph sequence is decomposed into a hyperstub
        % sequence using the multinomial distribution, M,
        % so that $H_i \in \mathbb{N}_0^{m\times N}$
        $H_i(j) = M(S_i(j), p_i)$,
    end
    % H_i' is a sequence of the true stub count
    $H_i' = H_i \cdot s_i$
    % Sum so that $H_i' \in \mathbb{N}_0^{1\times N}$
    $H_i'(j) = \sum_{\alpha=1}^{m} H_i'(\alpha, j)$
end
while *elements of each* H_i *are non-zero* **do**
    % Find the largest subgraph degree,
    $h_i(j) = \max\{\max\{H_1'\}, \max\{H_2'\}, \cdots, \max\{H_l'\}\}$
    % i.e., the j^{th} element of H_i.
    % Find all elements of the degree sequence at least this large and
    % select an element from d' at random
    $d' = \{d \in D : d \geq m\}$, $\delta = d'(random)$
    % pair $H_i(j)$ to δ and update
    % δ's available degree and H_i
    $\delta = \delta - H_i(j)$, $H_i(j) = 0$
end

References

1. Girvan, M. & Newman, M. E. J. (2002) Community structure in social and biological networks. *Proc. Nat. Acad. Sci.*, **99**, 7821–7826.
2. Newman, M. E. J. (2003) The structure and function of complex networks. *SIAM Rev.*, **45**, 167–256.
3. Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci.*, **79**, 2554–2558.
4. Ronen, S., Gonçalves, B., Hu, K. Z., Vespignani, A., Pinker, S. & Hidalgo, C. A. (2014) Links that speak: the global language network and its association with global fame. *Proc. Nat. Acad. Sci.*, **111**, E5616–E5622.
5. Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H. & Ratti, C. (2014) Quantifying the benefits of vehicle pooling with shareability networks. *Proc. Nat. Acad. Sci.*, **111**, 13290–13294.
6. Watts, D. J. & Strogatz, S. H. (1998) Collective dynamics of 'small-world' networks. *Nature*, **393**, 440–442.
7. Newman, M. E. J., Strogatz, S. H. & Watts, D. J. (2001) Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, **64**, 026118.
8. Kim, B. J. (2004) Performance of networks of artificial neurons: the role of clustering. *Phys. Rev. E*, **69**, 045101.
9. Volz, E. (2004) Random networks with tunable degree distribution and clustering. *Phys. Rev. E*, **70**, 056115.
10. Serrano, M. A. & Boguná, M. (2005) Tuning clustering in random networks with arbitrary degree distributions. *Phys. Rev. E*, **72**, 036133.
11. Bansal, S., Khandelwal, S. & Meyers, L. (2009) Exploring biological network structure with clustered random networks. *BMC Bioinformatics*, **10**, 405.
12. Ritchie, M., Berthouze, L., House, T. & Kiss, I. Z. (2014) Higher-order structure and epidemic dynamics in clustered networks. *J. Theor. Biol.*, **348**, 21–32.
13. Ritchie, M., Berthouze, L. & Kiss, I. Z. (2016) Beyond clustering: mean-field dynamics on networks with arbitrary subgraph composition. *Journal of Mathematical Biology*, **72**, 255–281.
14. Overbury, P. & Berthouze, L. (2015) Using novelty-biased GA to sample diversity in graphs satisfying constraints. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, ACM, pp. 1445–1446.
15. Barabási, A.-L. & Albert, R. (1999) Emergence of scaling in random networks. *Sci.*, **286**, 509–512.
16. Del Genio, C. I., Kim, H., Toroczkai, Z. & Bassler, K. E. (2010) Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS One*, **5**, 1–7.
17. Newman, M. E. J. (2002) Assortative mixing in networks. *Phys. Rev. Lett.*, **89**, 208701.
18. Bassler, K. E., Del Genio, C. I., Erdõs, P. L., Miklós, I. & Toroczkai, Z. (2015) Exact sampling of graphs with prescribed degree correlations. *New J. Phys.*, **17**, 083052.
19. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. & Alon, U. (2002) Network motifs: simple building blocks of complex networks. *Sci.*, **298**, 824–827.
20. Newman, M. E. J. (2003) Properties of highly clustered networks. *Phys. Rev. E*, **68**, 026121.
21. Karrer, B. & Newman, M. E. J. (2010) Random graphs containing arbitrary distributions of subgraphs. *Phys. Rev. E*, **82**, 066118.
22. Bollobás, B. (1980) A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *Eur. J. Combinatorics*, **1**, 311–316.
23. Miller, J. C. (2009) Percolation and epidemics in random clustered networks. *Phys. Rev. E*, **80**, 020901.
24. Newman, M. E. J. (2009) Random graphs with clustering. *Phys. Rev. Lett.*, **103**, 058701.
25. Milo, R., Kashtan, N., Itzkovitz, S., Newman, M. E. J. & Alon, U. (2003) On the uniform generation of random graphs with prescribed degree sequences. arXiv preprint cond-mat/0312028.
26. Klein-Hennig, H. & Hartmann, A. K. (2012) Bias in generation of random graphs. *Phys. Rev. E*, **85**, 026101.
27. House, T. & Keeling, M. J. (2010) The impact of contact tracing in clustered populations. *PLoS Comput. Biol.*, **6**, e1000721–e1000721.
28. House, T. & Keeling, M. J. (2011) Insights from unifying modern approximations to infections on networks. *J. Roy. Soc. Interface*, **8**, 67–73.

29. GREEN, D. M. & KISS, I. Z. (2010) Large-scale properties of clustered networks: implications for disease dynamics. *J. Biol. Dynamics*, **4**, 431–445.
30. MILLER, J. C. (2015) Complex contagions and hybrid phase transitions. *J. Complex Netw.*, **4**, 201–223.
31. O'SULLIVAN, D. J. P., O'KEEFFE, G. J., FENNELL, P. G. & GLEESON, J. P. (2015) Mathematical modelling of complex contagion on clustered networks. *Front. Phys.*, **3**, 71.
32. NEWMAN, M. (2010) *Networks: An Introduction*. New York: Oxford University Press, 2010.
33. HOUSE, T. (2010) Generalised network clustering and its dynamical implications. *Adv. Complex Sys.*, **13**, 281–291.
34. HOUSE, T., DAVIES, G., DANON, L. & KEELING, M. J. (2009) A motif-based approach to network epidemics. *Bull. Math. Biol.*, **71**, 1693–1706.
35. XULVI-BRUNET, R. & SOKOLOV, I. M. (2004) Reshuffling scale-free networks: From random to assortative. *Phys. Rev. E*, **70**, 066102.
36. KEELING, M. J. (1999) The effects of local spatial structure on epidemiological invasions. *Proc. Roy. Soc. London B*, **266**, 859–867.
37. KISS, I. Z., GREEN, D. M. & KAO, R. R. (2008) The effect of network mixing patterns on epidemic dynamics and the efficacy of disease contact tracing. *J. Roy. Soc. Interface*, **5**, 791–799.
38. MILLER, J. C. (2015) Personal communication.